Constraint-Based Sentence Compression: An Integer Programming Approach

Constraint-based Sentence Compression
An Integer Programming Approach

James Clarke and **Mirella Lapata**
School of Informatics, University of Edinburgh
2 Bucclecuch Place, Edinburgh EH8 9LW, UK
jclarke@ed.ac.uk, mlap@inf.ed.ac.uk

Abstract

The ability to compress sentences while preserving their grammaticality and most of their meaning has recently received much attention. Our work views sentence compression as an optimisation problem. We develop an integer programming formulation and infer globally optimal compressions in the face of linguistically motivated constraints. We show that such a formulation allows for relatively simple and knowledge-lean compression models that do not require parallel corpora or large-scale resources. The proposed approach yields results comparable and in some cases superior to state-of-the-art.

1 Introduction

A mechanism for automatically compressing sentences while preserving their grammaticality and most important information would greatly benefit a wide range of applications. Examples include text summarisation (Jing 2000), subtitle generation from spoken transcripts (Vandeghinste and Pan 2004) and information retrieval (Olivers and Dolan 1999). Sentence compression is a complex paraphrasing task with information loss involving substitution, deletion, insertion, and reordering operations. Recent years have witnessed increased interest on a simpler instantiation of the compression problem, namely word deletion (Knight and Marcu 2002; Riezler et al. 2003; Turner and Charniak 2005). More formally, given an input sentence of words $W = w_1, w_2, \ldots, w_n$, a compression is formed by removing any subset of these words.

Sentence compression has received both generative and discriminative formulations in the literature. Generative approaches (Knight and Marcu 2002; Turner and Charniak 2005) are instantiations of the noisy-channel model: given a long sentence l, the aim is to find the corresponding short sentence s which maximises the conditional probability $P(s|l)$. In a discriminative setting (Knight and Marcu 2002; Riezler et al. 2003; McDonald 2006), sentences are represented by a rich feature space (typically induced from parse trees) and the goal is to learn rewrite rules indicating which words should be deleted in a given context. Both modelling paradigms assume access to a training corpus consisting of original sentences and their compressions.

Unsupervised approaches to the compression problem are few and far between (see Hori and Furui 2004 and Turner and Charniak 2005 for exceptions). This is surprising considering that parallel corpora of original-compressed sentences are not naturally available in the way multilingual corpora are. The scarcity of such data is demonstrated by the fact that most work to date has focused on a single parallel corpus, namely the Ziff-Davis corpus (Knight and Marcu 2002). And some effort into developing appropriate training data would be necessary when porting existing algorithms to new languages or domains.

In this paper we present an unsupervised model of sentence compression that does not rely on a parallel corpus – all that is required is a corpus of uncompressed sentences and a parser. Given a long sentence, our task is to form a compression by preserving the words that maximise a scoring function. In our case, the scoring function is an n-gram language model, "with a few strings attached". While straightforward to estimate, a language model is a fairly primitive scoring function: it has no notion of the overall sentence structure, grammaticality or underlying meaning. We thus couple our language model with a small number of structural and semantic constraints capturing global properties of the compression process.

We encode the language model and linguistic constraints as linear inequalities and use Integer Programming (IP) to infer compressions that are consistent with both. The IP formulation allows us to capture global sentence properties and can be easily manipulated to provide compressions tailored for specific applications. For example, we

Proceedings of the COLING/ACL 2006 Main Conference Poster Sessions, pages 144–151,
Sydney, July 2006.

could prevent overly long or overly short compressions or generally avoid compressions that lack a main verb or consist of repetitions of the same word.

In the following section we provide an overview of previous approaches to sentence compression. In Section 3 we motivate the treatment of sentence compression as an optimisation problem and formulate our language model and constraints in the IP framework. Section 4 discusses our experimental set-up and Section 5 presents our results. Discussion of future work concludes the paper.

2 Previous Work

Jing (2000) was perhaps the first to tackle the sentence compression problem. Her approach uses multiple knowledge sources to determine which phrases in a sentence to remove. Central to her system is a grammar checking module that specifies which sentential constituents are grammatically obligatory and should therefore be present in the compression. This is achieved using simple rules and a large-scale lexicon. Other knowledge sources include WordNet and corpus evidence gathered from a parallel corpus of original-compressed sentence pairs. A phrase is removed only if it is not grammatically obligatory, not the focus of the local context and has a reasonable deletion probability (estimated from the parallel corpus).

In contrast to Jing (2000), the bulk of the research on sentence compression relies exclusively on corpus data for modelling the compression process without recourse to extensive knowledge sources (e.g., WordNet). Approaches based on the noisy-channel model (Knight and Marcu 2002; Turner and Charniak 2005) consist of a source model $P(s)$ (whose role is to guarantee that the generated compression is grammatical), a channel model $P(l|s)$ (capturing the probability that the long sentence l is an expansion of the compressed sentence s), and a decoder (which searches for the compression s that maximises $P(s)P(l|s)$). The channel model is typically estimated using a parallel corpus, although Turner and Charniak (2005) also present semi-supervised and unsupervised variants of the channel model that estimate $P(l|s)$ without parallel data.

Discriminative formulations of the compression task include decision-tree learning (Knight and Marcu 2002), maximum entropy (Riezler et al. 2003), support vector machines (Nguyen et al. 2004), and large-margin learning (McDonald 2006). We describe here the decision-tree model in more detail since we will use it as a basis for comparison when evaluating our own models (see Section 4). According to this model, compression is performed through a tree rewriting process inspired by the shift-reduce parsing paradigm. A sequence of shift-reduce-drop actions are performed on a long parse tree, l, to create a smaller tree, s.

The compression process begins with an input list generated from the leaves of the original sentence's parse tree and an empty stack. 'Shift' operations move leaves from the input list to the stack while 'drop' operations delete from the input list. Reduce operations are used to build trees from the leaves on the stack. A decision-tree is trained on a set of automatically generated learning cases from a parallel corpus. Each learning case has a target action associated with it and is decomposed into a set of indicative features. The decision-tree learns which action to perform given this set of features. The final model is applied in a deterministic fashion in which the features for the current state are extracted and the decision-tree is queried. This is repeated until the input list is empty and the final compression is recovered by traversing the leaves of resulting tree on the stack.

While most compression models operate over constituents, Hori and Furui (2004) propose a model which generates compressions through word deletion. The model does not utilise parallel data or syntactic information in any form. Given a prespecified compression rate, it searches for the compression with the highest score according to a function measuring the importance of each word and the linguistic likelihood of the resulting compressions (language model probability). The score is maximised through a dynamic programming algorithm.

Although sentence compression has not been explicitly formulated as an optimisation problem, previous approaches have treated it in these terms. The decoding process in the noisy-channel model searches for the best compression given the source and channel models. However, the compression found is usually sub-optimal as heuristics are used to reduce the search space or is only locally optimal due to the search method employed. The decoding process used in Turner and Charniak's (2005) model first searches for the best combination of rules to apply. As they traverse their list of compression rules they remove sentences outside the 100 best compressions (according to their channel model). This list is eventually truncated to 25 compressions.

In other models (Hori and Furui 2004; McDonald 2006) the compression score is maximised

using dynamic programming. The latter guarantees we will find the global optimum provided the principle of optimality holds. This principle states that given the current state, the optimal decision for each of the remaining stages does not depend on previously reached stages or previously made decisions (Winston and Venkataramanan 2003). However, we know this to be false in the case of sentence compression. For example, if we have included modifiers to the left of a head noun in the compression then it makes sense that we must include the head also. With a dynamic programming approach we cannot easily guarantee such constraints hold.

3 Problem Formulation

Our work models sentence compression explicitly as an optimisation problem. There are 2^n possible compressions for each sentence and while many of these will be unreasonable (Knight and Marcu 2002), it is unlikely that only one compression will be satisfactory. Ideally, we require a function that captures the operations (or rules) that can be performed on a sentence to create a compression while at the same time factoring how desirable each operation makes the resulting compression. We can then perform a search over *all* possible compressions and select the best one, as determined by how desirable it is.

Our formulation consists of two basic components: a language model (scoring function) and a small number of constraints ensuring that the resulting compressions are structurally and semantically valid. Our task is to find a globally optimal compression in the presence of these constraints. We solve this inference problem using Integer Programming without resorting to heuristics or approximations during the decoding process. Integer programming has been recently applied to several classification tasks, including relation extraction (Roth and Yih 2004), semantic role labelling (Punyakanok et al. 2004), and the generation of route directions (Marciniak and Strube 2005).

Before describing our model in detail, we introduce some of the concepts and terms used in Linear Programming and Integer Programming (see Winston and Venkataramanan 2003 for an introduction). Linear Programming (LP) is a tool for solving optimisation problems in which the aim is to maximise (or minimise) a given function with respect to a set of *constraints*. The function to be maximised (or minimised) is referred to as the *objective function*. Both the objective function and constraints must be linear. A number of *decision variables* are under our control which exert influence on the objective function. Specifically, they have to be optimised in order to maximise (or minimise) the objective function. Finally, a set of constraints restrict the values that the decision variables can take. Integer Programming is an extension of linear programming where all decision variables must take integer values.

3.1 Language Model

Assume we have a sentence $W = w_1, w_2, \ldots, w_n$ for which we wish to generate a compression. We introduce a decision variable for each word in the original sentence and constrain it to be binary; a value of 0 represents a word being dropped, whereas a value of 1 includes the word in the compression. Let:

$$y_i = \begin{cases} 1 & \text{if } w_i \text{ is in the compression} \\ 0 & \text{otherwise} \end{cases} \quad \forall i \in [1 \ldots n]$$

If we were using a unigram language model, our objective function would maximise the overall sum of the decision variables (i.e., words) multiplied by their unigram probabilities (all probabilities throughout this paper are log-transformed):

$$\max z = \sum_{i=1}^{n} y_i \cdot P(w_i)$$

Thus if a word is selected, its corresponding y_i is given a value of 1, and its probability $P(w_i)$ according to the language model will be counted in our total score, z.

A unigram language model will probably generate many ungrammatical compressions. We therefore use a more context-aware model in our objective function, namely a trigram model. Formulating a trigram model in terms of an integer program becomes a more involved task since we now must make decisions based on word sequences rather than isolated words. We first create some extra decision variables:

$$p_i = \begin{cases} 1 \text{ if } w_i \text{ starts the compression} \\ 0 \text{ otherwise} \end{cases} \forall i \in [1 \ldots n]$$

$$q_{ij} = \begin{cases} 1 \text{ if sequence } w_i, w_j \text{ ends} \\ \quad \text{the compression} & \forall i \in [1 \ldots n-1] \\ 0 \text{ otherwise} & \forall j \in [i+1 \ldots n] \end{cases}$$

$$x_{ijk} = \begin{cases} 1 \text{ if sequence } w_i, w_j, w_k & \forall i \in [1 \ldots n-2] \\ \quad \text{is in the compression} & \forall j \in [i+1 \ldots n-1] \\ 0 \text{ otherwise} & \forall k \in [j+1 \ldots n] \end{cases}$$

Our objective function is given in Equation (1). This is the sum of all possible trigrams that can occur in all compressions of the original sentence where w_0 represents the 'start' token and w_i is the ith word in sentence W. Equation (2) constrains

the decision variables to be binary.

$$\begin{aligned}\max z \;=\; & \sum_{i=1}^{n} p_i \cdot P(w_i|\text{start}) \\ & + \sum_{i=1}^{n-2}\sum_{j=i+1}^{n-1}\sum_{k=j+1}^{n} x_{ijk} \cdot P(w_k|w_i,w_j) \\ & + \sum_{i=0}^{n-1}\sum_{j=i+1}^{n} q_{ij} \cdot P(\text{end}|w_i,w_j) \quad (1)\end{aligned}$$

subject to:

$$y_i, p_i, q_{ij}, x_{ijk} = 0 \text{ or } 1 \quad (2)$$

The objective function in (1) allows any combination of trigrams to be selected. This means that invalid trigram sequences (e.g., two or more trigrams containing the symbol 'end') could appear in the output compression. We avoid this situation by introducing *sequential constraints* (on the decision variables y_i, x_{ijk}, p_i, and q_{ij}) that restrict the set of allowable trigram combinations.

Constraint 1 Exactly one word can begin a sentence.

$$\sum_{i=1}^{n} p_i = 1 \quad (3)$$

Constraint 2 If a word is included in the sentence it must either start the sentence or be preceded by two other words or one other word and the 'start' token w_0.

$$y_k - p_k - \sum_{i=0}^{k-2}\sum_{j=1}^{k-1} x_{ijk} = 0 \quad (4)$$
$$\forall k : k \in [1 \ldots n]$$

Constraint 3 If a word is included in the sentence it must either be preceded by one word and followed by another or it must be preceded by one word and end the sentence.

$$y_j - \sum_{i=0}^{j-1}\sum_{k=j+1}^{n} x_{ijk} - \sum_{i=0}^{j-1} q_{ij} = 0 \quad (5)$$
$$\forall j : j \in [1 \ldots n]$$

Constraint 4 If a word is in the sentence it must be followed by two words or followed by one word and then the end of the sentence or it must be preceded by one word and end the sentence.

$$y_i - \sum_{j=i+1}^{n-1}\sum_{k=j+1}^{n} x_{ijk} - \sum_{j=i+1}^{n} q_{ij} - \sum_{h=0}^{i-1} q_{hi} = 0 \quad (6)$$
$$\forall i : i \in [1 \ldots n]$$

Constraint 5 Exactly one word pair can end the sentence.

$$\sum_{i=0}^{n-1}\sum_{j=i+1}^{n} q_{ij} = 1 \quad (7)$$

Example compressions using the trigram model just described are given in Table 1. The model in its current state does a reasonable job of modelling local word dependencies, but is unable to capture syntactic dependencies that could potentially allow more meaningful compressions. For example, it does not know that *Pasok Party* is the object of *founded* or that *Appleshare* modifies *Printer Server*.

O:	He became a power player in Greek Politics in 1974, when he founded the socialist Pasok Party.
LM:	He became a player in the Pasok.
Mod:	He became a player in the Pasok Party.
Sen:	He became a player in politics.
Sig:	He became a player in politics when he founded the Pasok Party.
O:	Finally, AppleShare Printer Server, formerly a separate package, is now bundled with AppleShare File Server.
LM:	Finally, AppleShare, a separate, AppleShare.
Mod:	Finally, AppleShare Server, is bundled.
Sen:	Finally, AppleShare Server, is bundled with Server.
Sig:	AppleShare Printer Server package is now bundled with AppleShare File Server.

Table 1: Compression examples (O: original sentence, LM: compression with the trigram model, Mod: compression with LM and modifier constraints, Sen: compression with LM, Mod and sentential constraints, Sig: compression with LM, Mod, Sen, and significance score)

3.2 Linguistic Constraints

In this section we propose a set of global constraints that extend the basic language model presented in Equations (1)–(7). Our aim is to bring some syntactic knowledge into the compression model and to preserve the meaning of the original sentence as much as possible. Our constraints are linguistically and semantically motivated in a similar fashion to the grammar checking component of Jing (2000). Importantly, we do not require any additional knowledge sources (such as a lexicon) beyond the parse and grammatical relations of the original sentence. This is provided in our experiments by the Robust Accurate Statistical Parsing (RASP) toolkit (Briscoe and Carroll 2002). However, there is nothing inherent in our formulation that restricts us to RASP; any other parser with similar output could serve our purposes.

Modifier Constraints Modifier constraints ensure that relationships between head words and their modifiers remain grammatical in the compression:

$$y_i - y_j \geq 0 \quad (8)$$
$$\forall i,j : w_j \in w_i\text{'s } \texttt{ncmods}$$
$$y_i - y_j \geq 0 \quad (9)$$
$$\forall i,j : w_j \in w_i\text{'s } \texttt{detmods}$$

Equation (8) guarantees that if we include a non-clausal modifier (ncmod) in the compression then the head of the modifier must also be included; this is repeated for determiners (detmod) in (9).

We also want to ensure that the meaning of the original sentence is preserved in the compression, particularly in the face of negation. Equation (10) implements this by forcing *not* in the compression when the head is included. A similar constraint is added for possessive modifiers (e.g., *his*, *our*), as shown in Equation (11). Genitives (e.g., *John's gift*) are treated separately, mainly because they are encoded as different relations in the parser (see Equation (12)).

$$y_i - y_j = 0 \quad (10)$$
$$\forall i, j : w_j \in w_i\text{'s ncmods} \wedge w_j = \text{not}$$
$$y_i - y_j = 0 \quad (11)$$
$$\forall i, j : w_j \in w_i\text{'s possessive detmods}$$
$$y_i - y_j = 0 \quad (12)$$
$$\forall i, j : w_i \in \text{possessive ncmods}$$
$$\wedge w_j = \text{possessive}$$

Compression examples with the addition of the modifier constraints are shown in Table 1. Although the compressions are grammatical (see the inclusion of *Party* due to the modifier *Pasok* and *Server* due to *AppleShare*), they are not entirely meaning preserving.

Sentential Constraints We also define a few intuitive constraints that take the overall sentence structure into account. The first constraint (Equation (13)) ensures that if a verb is present in the compression then so are its arguments, and if any of the arguments are included in the compression then the verb must also be included. We thus force the program to make the same decision on the verb, its subject, and object.

$$y_i - y_j = 0 \quad (13)$$
$$\forall i, j : w_j \in \text{subject/object of verb } w_i$$

Our second constraint forces the compression to contain at least one verb provided the original sentence contains one as well:

$$\sum_{i \in verbs} y_i \geq 1 \quad (14)$$

Other sentential constraints include Equations (15) and (16) which apply to prepositional phrases, wh-phrases and complements. These constraints force the introducing term (i.e., the preposition, complement or wh-word) to be included in the compression if any word from within the syntactic constituent is also included. The reverse is also true, i.e., if the introducing term is included at least one other word from the syntactic constituent should also be included.

$$y_i - y_j \geq 0 \quad (15)$$
$$\forall i, j : w_j \in \text{PP/COMP/WH-P}$$
$$\wedge w_i \text{ starts PP/COMP/WH-P}$$
$$\sum_{i \in \text{PP/COMP/WH-P}} y_i - y_j \geq 0 \quad (16)$$
$$\forall j : w_j \text{ starts PP/COMP/WH-P}$$

We also wish to handle coordination. If two head words are conjoined in the original sentence, then if they are included in the compression the coordinating conjunction must also be included:

$$(1 - y_i) + y_j \geq 1 \quad (17)$$
$$(1 - y_i) + y_k \geq 1 \quad (18)$$
$$y_i + (1 - y_j) + (1 - y_k) \geq 1 \quad (19)$$
$$\forall i, j, k : w_j \wedge w_k \text{ conjoined by } w_i$$

Table 1 illustrates the compression output when sentential constraints are added to the model. We see that *politics* is forced into the compression due to the presence of *in*; furthermore, since *bundled* is in the compression, its object *with Server* is included too.

Compression-related Constraints Finally, we impose some hard constraints on the compression output. First, Equation (20) disallows anything within brackets in the original sentence from being included in the compression. This is a somewhat superficial attempt at excluding parenthetical and potentially unimportant material from the compression. Second, Equation (21) forces personal pronouns to be included in the compression. The constraint is important for generating coherent document as opposed to sentence compressions.

$$y_i = 0 \quad (20)$$
$$\forall i : w_i \in \text{brackets}$$
$$y_i = 1 \quad (21)$$
$$\forall i : w_i \in \text{personal pronouns}$$

It is also possible to influence the length of the compressed sentence. For example, Equation (22) forces the compression to contain at least b tokens. Alternatively, we could force the compression to be exactly b tokens (by substituting $\geq$ with $=$ in (22)) or to be less than b tokens (by replacing $\geq$ with $\leq$).[1]

$$\sum_{i=1}^{n} y_i \geq b \quad (22)$$

3.3 Significance Score

While the constraint-based language model produces more grammatical output than a regular lan-

[1] Compression rate can be also limited to a range by including two inequality constraints.

guage model, the sentences are typically not great compressions. The language model has no notion of which content words to include in the compression and thus prefers words it has seen before. But words or constituents will be of different relative importance in different documents or even sentences.

Inspired by Hori and Furui (2004), we add to our objective function (see Equation (1)) a significance score designed to highlight important content words. Specifically, we modify Hori and Furui's significance score to give more weight to content words that appear in the deepest level of embedding in the syntactic tree. The latter usually contains the gist of the original sentence:

$$I(w_i) = \frac{l}{N} \cdot f_i \log \frac{F_a}{F_i} \quad (23)$$

The significance score above is computed using a large corpus where w_i is a topic word (i.e., a noun or verb), f_i and F_i are the frequency of w_i in the document and corpus respectively, and F_a is the sum of all topic words in the corpus. l is the number of clause constituents above w_i, and N is the deepest level of embedding. The modified objective function is given below:

$$\begin{aligned} \max z \quad = \quad & \sum_{i=1}^{n} y_i \cdot I(w_i) + \sum_{i=1}^{n} p_i \cdot P(w_i|\text{start}) \\ & + \sum_{i=1}^{n-2} \sum_{j=i+1}^{n-1} \sum_{k=j+1}^{n} x_{ijk} \cdot P(w_k|w_i, w_j) \\ & + \sum_{i=0}^{n-1} \sum_{j=i+1}^{n} q_{ij} \cdot P(\text{end}|w_i, w_j) \quad (24) \end{aligned}$$

A weighting factor could be also added to the objective function, to counterbalance the importance of the language model and the significance score.

4 Evaluation Set-up

We evaluated the approach presented in the previous sections against Knight and Marcu's (2002) decision-tree model. This model is a good basis for comparison as it operates on parse trees and therefore is aware of syntactic structure (as our models are) but requires a large parallel corpus for training whereas our models do not; and it yields comparable performance to the noisy-channel model.[2] The decision-tree model was compared against two variants of our IP model. Both variants employed the constraints described in Section 3.2 but differed in that one variant included the significance score in its objective function (see (24)), whereas the other one did not (see (1)). In both cases the sequential constraints from Section 3.1 were applied to ensure that the language model was well-formed. We give details below on the corpora we used and explain how the different model parameters were estimated. We also discuss how evaluation was carried out using human judgements.

Corpora We evaluate our systems on two different corpora. The first is the compression corpus of Knight and Marcu (2002) derived automatically from document-abstract pairs of the Ziff-Davis corpus. This corpus has been used in most previous compression work. We also created a compression corpus from the HUB-4 1996 English Broadcast News corpus (provided by the LDC). We asked annotators to produce compressions for 50 broadcast news stories (1,370 sentences).[3]

The Ziff-Davis corpus is partitioned into training (1,035 sentences) and test set (32 sentences). We held out 50 sentences from the training for development purposes. We also split the Broadcast News corpus into a training and test set (1,237/133 sentences). Forty sentences were randomly selected for evaluation purposes, 20 from the test portion of the Ziff-Davis corpus and 20 from the Broadcast News corpus test set.

Parameter Estimation The decision-tree model was trained, using the same feature set as Knight and Marcu (2002) on the Ziff-Davis corpus and used to obtain compressions for both test corpora.[4] For our IP models, we used a language model trained on 25 million tokens from the North American News corpus using the CMU-Cambridge Language Modeling Toolkit (Clarkson and Rosenfeld 1997) with a vocabulary size of 50,000 tokens and Good-Turing discounting. The significance score used in our second model was calculated using 25 million tokens from the Broadcast News Corpus (for the spoken data) and 25 million tokens from the American News Text Corpus (for the written data). Finally, the model that includes the significance score was optimised against a loss function similar to McDonald (2006) to bring the language model and the score into harmony. We used Powell's method (Press et al. 1992) and 50 sentences (randomly selected from the training set).

[2] Turner and Charniak (2005) argue that the noisy-channel model is not an appropriate compression model since it uses a source model trained on uncompressed sentences and as a result tends to consider compressed sentences less likely than uncompressed ones.

[3] The corpus is available from `http://homepages.inf.ed.ac.uk/s0460084/data/`.

[4] We found that the decision-tree was unable to produce meaningful compressions when trained on the Broadcast News corpus (in most cases it recreated the original sentence). Thus we used the decision model trained on Ziff-Davis to generate Broadcast News compressions.

We also set a minimum compression length (using the constraint in Equation (22)) in both our models to avoid overly short compressions. The length was set at 40% of the original sentence length or five tokens, whichever was larger. Sentences under five tokens were not compressed.

In our modeling framework, we generate and solve an IP for every sentence we wish to compress. We employed *lp_solve* for this purpose, an efficient Mixed Integer Programming solver.[5] Sentences typically take less than a few seconds to compress on a 2 GHz Pentium IV machine.

Human Evaluation As mentioned earlier, the output of our models is evaluated on 40 examples. Although the size of our test set is comparable to previous studies (which are typically assessed on 32 sentences from the Ziff-Davis corpus), the sample is too small to conduct significance testing. To counteract this, human judgements are often collected on compression output; however the evaluations are limited to small subject pools (often four judges; Knight and Marcu 2002; Turner and Charniak 2005; McDonald 2006) which makes difficult to apply inferential statistics on the data. We overcome this problem by conducting our evaluation using a larger sample of subjects.

Specifically, we elicited human judgements from 56 unpaid volunteers, all self reported native English speakers. The elicitation study was conducted over the Internet. Participants were presented with a set of instructions that explained the sentence compression task with examples. They were asked to judge 160 compressions in total. These included the output of the three automatic systems on the 40 test sentences paired with their gold standard compressions. Participants were asked to read the original sentence and then reveal its compression by pressing a button. They were told that all compressions were generated automatically. A Latin square design ensured that subjects did not see two different compressions of the same sentence. The order of the sentences was randomised. Participants rated each compression on a five point scale based on the information retained and its grammaticality. Examples of our experimental items are given in Table 2.

O:	Apparently Fergie very much wants to have a career in television.
G:	Fergie wants a career in television.
D:	A career in television.
LM:	Fergie wants to have a career.
Sig:	Fergie wants to have a career in television.
O:	The SCAMP module, designed and built by Unisys and based on an Intel process, contains the entire 48-bit A-series processor.
G:	The SCAMP module contains the entire 48-bit A-series processor.
D:	The SCAMP module designed Unisys and based on an Intel process.
LM:	The SCAMP module, contains the 48-bit A-series processor.
Sig:	The SCAMP module, designed and built by Unisys and based on process, contains the A-series processor.

Table 2: Compression examples (O: original sentence, G: Gold standard, D: Decision-tree, LM: IP language model, Sig: IP language model with significance score)

Model	CompR	Rating
Decision-tree	56.1%	$2.22^{*\dagger}$
LangModel	49.0%	$2.23^{*\dagger}$
LangModel+Significance	73.6%	2.83^{*}
Gold Standard	62.3%	$3.68^{\dagger}$

Table 3: Compression results; compression rate (CompR) and average human judgements (Rating); *: sig. diff. from gold standard; †: sig. diff. from LangModel+Significance

5 Results

Our results are summarised in Table 3 which details the compression rates[6] and average human ratings (Rating) for the three systems and the gold standard. As can be seen, the IP language model (LangModel) is most aggressive in terms of compression rate as it reduces the original sentences on average by half (49%). Recall that we enforce a minimum compression rate of 40% (see (22)). The fact that the resulting compressions are longer, indicates that our constraints instill some linguistic knowledge into the language model, thus enabling it to prefer longer sentences over extremely short ones. The decision-tree model compresses slightly less than our IP language model at 56.1% but still below the gold standard rate. We see a large compression rate increase from 49% to 73.6% when we introduce the significance score into the objective function. This is around 10% higher than the gold standard compression rate.

We now turn to the results of our elicitation study. We performed an Analysis of Variance (ANOVA) to examine the effect of different system compressions. Statistical tests were carried out on the mean of the ratings shown in Table 3. We observe a reliable effect of compression type by sub-

[5]The software is available from `http://www.geocities.com/lpsolve/`.

[6]We follow previous work (see references) in using the term "compression rate" to refer to the percentage of words *retained* in the compression.

jects ($F_1(3,165) = 132.74,\ p < 0.01$) and items ($F_2(3,117) = 18.94,\ p < 0.01$). Post-hoc Tukey tests revealed that gold standard compressions are perceived as significantly better than those generated by all automatic systems ($\alpha < 0.05$). There is no significant difference between the IP language model and decision-tree systems. However, the IP model with the significance score delivers a significant increase in performance over the language model and the decision tree ($\alpha < 0.05$).

These results indicate that reasonable compressions can be obtained with very little supervision. Our constraint-based language model does not make use of a parallel corpus, whereas our second variant uses only 50 parallel sentences for tuning the weights of the objective function. The models described in this paper could be easily adapted to other domains or languages provided that syntactic analysis tools are to some extent available.

6 Conclusions and Future Work

In this paper we have presented a novel method for automatic sentence compression. A key aspect of our approach is the use of integer programming for inferring globally optimal compressions in the presence of linguistically motivated constraints. We have shown that such a formulation allows for a relatively simple and knowledge-lean compression model that does not require parallel corpora or access to large-scale knowledge bases. Our results demonstrate that the IP model yields performance comparable to state-of-the-art without any supervision. We also observe significant performance gains when a small amount of training data is employed (50 parallel sentences). Beyond the systems discussed in this paper, the approach holds promise for other models using decoding algorithms for searching the space of possible compressions. The search process could be framed as an integer program in a similar fashion to our work here.

We obtain our best results using a model whose objective function includes a significance score. The significance score relies mainly on syntactic and lexical information for determining whether a word is important or not. An appealing future direction is the incorporation of discourse-based constraints into our models. The latter would highlight topical words at the document-level instead of considering each sentence in isolation. Another important issue concerns the portability of the models presented here to other languages and domains. We plan to apply our method to languages with more flexible word order than English (e.g., German) and more challenging spoken domains (e.g., meeting data) where parsing technology may be less reliable.

Acknowledgements

Thanks to Jean Carletta, Amit Dubey, Frank Keller, Steve Renals, and Sebastian Riedel for helpful comments and suggestions. Lapata acknowledges the support of EPSRC (grant GR/T04540/01).

References

Briscoe, E. J. and J. Carroll. 2002. Robust accurate statistical annotation of general text. In *Proceedings of the 3rd LREC*. Las Palmas, Gran Canaria, pages 1499–1504.

Clarkson, Philip and Ronald Rosenfeld. 1997. Statistical language modeling using the CMU–cambridge toolkit. In *Proceedings of Eurospeech*. Rhodes, Greece, pages 2707–2710.

Hori, Chiori and Sadaoki Furui. 2004. Speech summarization: an approach through word extraction and a method for evaluation. *IEICE Transactions on Information and Systems* E87-D(1):15–25.

Jing, Hongyan. 2000. Sentence reduction for automatic text summarization. In *Proceedings of the 6th ANLP*. Seattle, WA, pages 310–315.

Knight, Kevin and Daniel Marcu. 2002. Summarization beyond sentence extraction: a probabilistic approach to sentence compression. *Artificial Intelligence* 139(1):91–107.

Marciniak, Tomasz and Michael Strube. 2005. Beyond the pipeline: Discrete optimization in NLP. In *Proceedings of the 9th CoNLL*. Ann Arbor, MI, pages 136–143.

McDonald, Ryan. 2006. Discriminative sentence compression with soft syntactic constraints. In *Proceedings of the 11th EACL*. Trento, Italy, pages 297–304.

Nguyen, Minh Le, Akira Shimazu, Susumu Horiguchi, Tu Bao Ho, and Masaru Fukushi. 2004. Probabilistic sentence reduction using support vector machines. In *Proceedings of the 20th COLING*. Geneva, Switzerland, pages 743–749.

Olivers, S. H. and W. B. Dolan. 1999. Less is more; eliminating index terms from subordinate clauses. In *Proceedings of the 37th ACL*. College Park, MD, pages 349–356.

Press, William H., Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. 1992. *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge University Press.

Punyakanok, Vasin, Dan Roth, Wen-tau Yih, and Dav Zimak. 2004. Semantic role labeling via integer linear programming inference. In *Proceedings of the 20th COLING*. Geneva, Switzerland, pages 1346–1352.

Riezler, Stefan, Tracy H. King, Richard Crouch, and Annie Zaenen. 2003. Statistical sentence condensation using ambiguity packing and stochastic disambiguation methods for lexical-functional grammar. In *Proceedings of the HLT/NAACL*. Edmonton, Canada, pages 118–125.

Roth, Dan and Wen-tau Yih. 2004. A linear programming formulation for global inference in natural language tasks. In *Proceedings of the 8th CoNLL*. Boston, MA, pages 1–8.

Turner, Jenine and Eugene Charniak. 2005. Supervised and unsupervised learning for sentence compression. In *Proceedings of the 43rd ACL*. Ann Arbor, MI, pages 290–297.

Vandeghinste, Vincent and Yi Pan. 2004. Sentence compression for automated subtitling: A hybrid approach. In *Proceedings of the ACL Workshop on Text Summarization*. Barcelona, Spain, pages 89–95.

Winston, Wayne L. and Munirpallam Venkataramanan. 2003. *Introduction to Mathematical Programming*. Brooks/Cole.